NagaSwap 프로토콜 결제 논리

앞서 언급한 전면 실행 및 충돌 문제를 해결하면서 완전히 분산되고 신뢰할 필요 없는 방식으로 유동성을 공유할 수 있도록 하기 위해 NagaSwap 프로토콜을 소개합니다. NagaSwap 프로토콜은 들어오는 주문의 공정한 순서를 설정하여 충돌 및 전면 실행을 해결하는 새로운 온체인 결제 논리 체계입니다. NagaSwap 프로토콜은 검증 가능한 지연 기능을 사용하고 공개적으로 검증 가능한 경과시간증명 구성을 사용하여 이를 수행하며, 이는 또한 re-레이어 간의 원활한 유동성 공유를 가능하게 합니다. 이 결제 논리는 프로토콜에 구애받지 않으므로 여러 DEX 프로토콜이 완전히 분산되고 신뢰할 필요 없는 방식으로 거래를 결제하고 유동성을 공유할 수 있습니다.

1. 예비 정의

정의 3.1. 시장은 메이크 오더와 테이크 오더로 구성됩니다:

- *M은* 메이크 오더*이며* M_{vol}는 *메이크 오더의 볼륨입니다.*
- *T는 주소 A로* 테이커 $T_{\&}$가 보낸 테이크 오더입니다. T'는$T_{\&}$의주문과 동일한메이크주문M에 대해 테이크주문을 제출하는주소B가있는 충돌하거나적대적인테이커입니다.또한이 테이크오더의볼륨으로T_{vol}를 정의합니다.

정의 3.2. 검증 가능한 지연 함수(VDF)의 단순화된 구현은 V = Setup(λ, t), Eval(ek, x), Verify(vk, x, y, π) 인 알고리즘의 투플입니다.

- Setup(λ) => pp = (ek, vk)는 보안 매개 변수 λ 를 취하여 평가 키 ek와 검증 키 vk로 구성된 공개 매개 변수 pp를 생성합니다. 실제로 pp는 네트워크 시작시 정적으로 유지됩니다. t는 프로토콜에서 증분되고 증명의 일부로 제출되므로 VDF의 원래 정의에 포함된 퍼즐 난이도 매개 변수 t를 생략하여 VDF의 정의를 단순화합니다.
- Eval(ek, x) => (y, π)는 입력 a를 받아 y와 증명 π를 생성합니다. 실제로 Eval은 평가와 검증 사이에 기하 급수적인 차이가 있는 결정론적 병렬화 하드 함수입니다.
- Verify(vk, x, y, π) = {True, False}는 입력 x, 출력 y를 취하고 불 방식으로 True 또는 False를 출력하는 결정론적 검증 알고리즘입니다. 실제로(특히 디코딩 가능한 속성을 충족하지 않는 VDF 후보의 경우) 알고리즘은 증명 π 도 평가합니다.

정의 3.3. aker는 VDF를 로컬에서 계산하고 다음을 결제 논리 스마트 계약에 지속적으로 제출합니다.

Commit(π, t, x, yt, vk, T, S,)

여기서 a는 검증 알고리즘에서 생성 된 증명의 목록입니다 (간단함을 위해 {π- | t ∈ N}을 π로 축약함), t는 테이커가 계산한 Eval의 반복 횟수, x는 테이크 오더 거래 정보 T를 인코딩하는 시작 입력, y-는 Eval의 t-0 반복의 출력, T는 테이커가 원하는 테이크 오더, S, = Sig(sk, m = nonce,-6||T))는 테이크 오더와 연결된 이전 블록의 nonce에 대한 개인 키 sk가 있는 테이커의 서명입니다.

정의 3.4. 결제 논리는 Commit을 수신하고 다음과 같이 평가합니다

Confirm(π, t, x, y, vk, T, S,)

Verify(vk, x, y, π) → {True, False} 를 사용하여 $T_{\&}$의 Commit이 유효한지 여부를 확인합니다. 주어진 블록 내에 주어진 메이크 오더 M을 받고자 하는 바람을 나타내는 n개의 Commit이 있다고 가정합니다. 결제 논리가 메이크 오더에 대해 하나 이상의 Commit을 수신하면 max{ t_7 | n ∈ N} 충돌하는 n개의 Commit 각각에서 얻은 t는 메이크 오더 M의 임시 승자로 기능합니다. 또한 δ 첫 번째 임시 승자와 영구 승자의 해결 사이의 라운드 (블록) 수의 지연으로 정의됩니다. δ 는 원하는대로 분산형 교환을 위해 매개 변수화 할 수 있으며 다음 섹션에서 더 자세히 설명합니다.

정의 3.5. 가장 간단한 경우 $M_{()^*} < T_{a(vol)} + T_{b(vol)}$ 가정에서 결합된 테이크 오더가 메이크 오더 볼륨보다 더 많은 볼륨을 갖는다고 가정하면 $T_{\&}$, T'가 충돌합니다. 둘 다 동일한 라운드 r 내에서 동일한 M에 대한 테이크 오더를 제출하고 결제 논리는 공정한 방식으로 우승 테이커를 결정론적으로 선택할 수 없습니다. Front-Running은 $T_{\&}$가 유효한 테이크 오더를 먼저 제출하지만 적대적인 T'가, T'이후에 $T_{\&}$가 전송 되었음에도 불구하고 $T_{\&}$가 확인되기 전에(일반적으로 단순히 더 높은 가스 요금을 사용하여 수행됨) M을 취하기 위해 M에 대해 동일한 테이크 오더를 제출할 때 발생합니다.

2. 경과 시간을 증명하는 검증 가능한 지연 기능

우리는 *확인 가능한 지연 함수*(VDF)를 경과 시간 증명을 위한 메커니즘으로 활용합니다. 여기서 받는 사람이 Commit에서 청구한 기간 동안 주문을 실제로 준수했는지 확인하는 결제 논리에 시간 증명을 제출할 수 있습니다. VDF에서 제안된 순차적이고 효율적으로 검증 가능한 특성을 충족하는 것 외에도, VDF 후보가 증분이어야 하므로 경도 매개 변수 t가 설정 대신 증명에 지정될 수 있습니다.

주어진 블록 r에서 테이크 오더 T를 제출하는 테이커의 경우 테이커의 시작 값 x를 다음과 같이 표시합니다:

$$x = H(Sig(sk, m))$$

여기서 m = nonce$_{-6}$||T (즉, 테이크 오더와 연결된 이전 블록의 nonce), sk는 $T_{\&}$의 비밀 키이고 Sig (sk, m)는 메시지 m에 대한 디지털 서명입니다. 단순화를 위해 S, = Sig (sk, m)를 표시합니다. 이 구조는 주어진 블록 r에서 x가 이전 블록 r-1의 nonce에 의존하기 때문에 미래 블록에 대해 x를 미리 계산할 수 없음을 보장합니다.

NagaSwap 프로토콜의 결제 논리의 경우 Eval 및 Verify에 대해 다음과 같은 네 가지 후보 VDF 알고리즘을 사용합니다: Sloth++, nagaswap 합리적인 맵의 순열 다항식 체인, Wesolowski의 Efficient VDF 및 Pietrzak의 Simple VDF. Boneh, Bonneau, Bunz, Fisch가 제안한 SNARK를 사용한 Sloth++ 또는 Permutation Polynomial Chain에서 f (t, x) = g-(x) 형식의 반복 순차 함수가 사용됩니다. g를 Eval 알고리즘으로 대체하고 Verify를 통합하면 다음과 같은 이점이 있습니다:

$$\textit{computed in parallel}\begin{cases} x = x_0 \to \underbrace{Eval(x_0)}_{x_1} \to \underbrace{Eval(Eval(x_0))}_{x_2} \to \ldots \to \underbrace{Eval^{(t)}(x_0)}_{x_t} = y_{t+1} \\ x = x_0 = \underbrace{Verify(x_1) \leftarrow x_1}_{\pi_1} = \underbrace{Verify(x_2) \leftarrow x_2}_{\pi_2} \leftarrow \ldots \leftarrow \underbrace{Verify(x_t) \leftarrow x_t}_{\pi_t} = y_{t+1} \end{cases}$$

Sloth++2의 경우, 주어진 $x \in Z_p^*$에 대해 간결한 검증을 위해 SNARK를 통합하는 Sloth 기능에 대한 개선 제안이 있습니다; 여기서 p는 $p \equiv 3 \bmod 4$, $Eval(x) = \sqrt{x} \bmod p = x^{p+1/4} \bmod p$ 가되는 충분히 큰 소수입니다. 검증을 위해 x mod p = Verify(y) = y^2, 도다음을 생성합니다 SNARKProve(ek,y) $\Rightarrow \pi$. nagaswap 합리적 맵의 순열 다항식 체인의 경우 F_p에 대해 Guralnick과 Muller가 제안한 후보 다항식을 사용합니다:

$$\frac{(x^8 - ax - a).(x^8 - ax + a)^8 + ((x^8 - ax + a)^2 + 4a^2x)^{(s+1)/2}}{2x^8}$$

여기서 s= p, 홀수 소수 p 및 일부 정수 r 및 a는 F>의 (s-1) 거듭 제곱이 아닙니다. Eval(x)는 난이도 매개 변수 s를 사용하여 다항식의 반전을 계산하는 알고리즘입니다. 이 제안은 다항식 GCD를 계산하는 것이 가장 빠른 반전 방법이며, 최소 s개의 병렬 프로세서가 있는 최적화된 하드웨어에서 평가되더라도 최소 s의 순차적 단계가 필요하다고 가정합니다. 그러나 최소 병렬 프로세서가 있는 최적화되지 않은 하드웨어는 O (s^F) 시간에 Eval을 평가합니다.

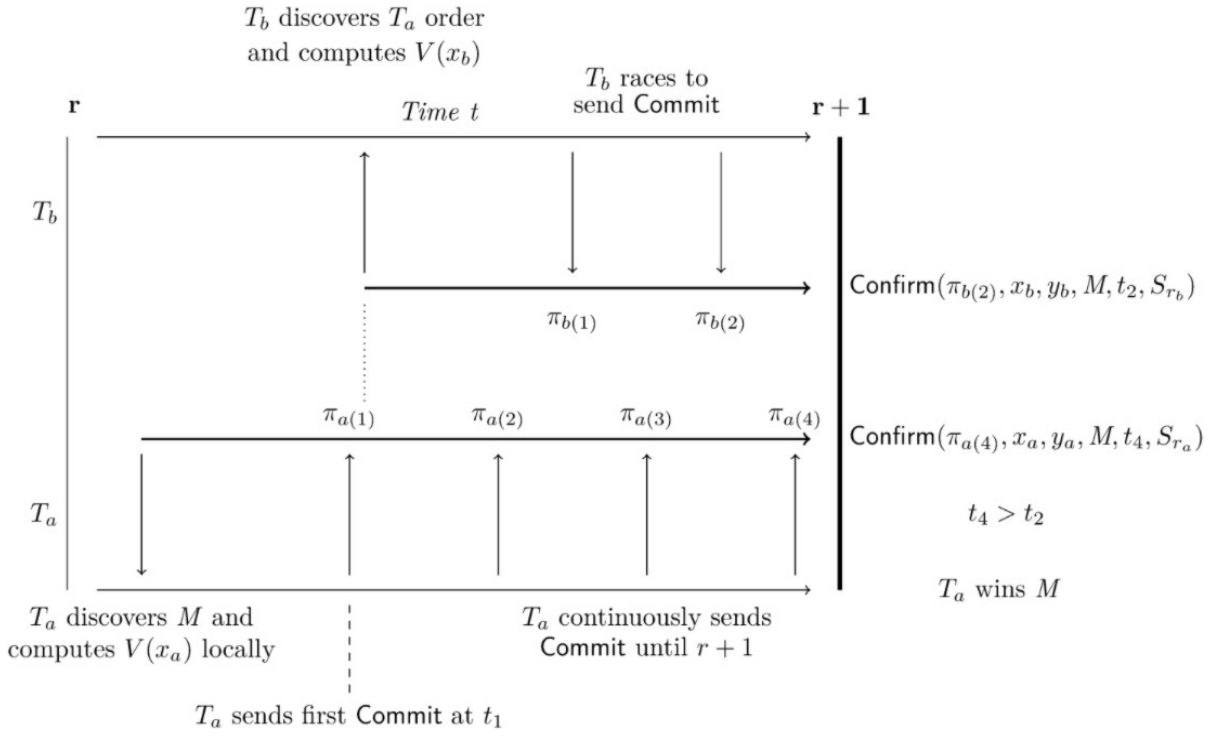
설정을 수정하면 순서를 알 수 없는 그룹 G의 트랩 도어를 기반으로 Rivest, Shamir 및 Wagner(RSW) 시간 잠금 퍼즐을 사용하는 Wesolowski의 구성을

구현할 수 있습니다. 난이도 매개 변수 t가 평가 중에 순차적으로 증가하기 때문에 RSW 구성에는 반복 순차 함수가 필요하지 않습니다. 그러나 이 구성은 RSW 알고리즘의 반전을 계산하는 효율적인 방법이 없기 때문에 이전 두 구성이 수행하는 디코딩 가능한 속성을 충족하지 않습니다. RSW는 $y = x^F \bmod N$이 주어지면 평가자가 N의 그룹 순서 또는 인수 분해를 알 수 없는 경우 y를 평가하기 위해 t 순차 제곱이 필요하다고 가정합니다. 이를 구현하기 위해 Wesolowski의 가상 2차 순서를 사용하여 알 수 없는 그룹 순서를 생성하여 설정을 수정합니다. 이렇게 하면 판별자가 클 때 RSW를 해결하는 것보다 클래스 그룹을 더 빨리 계산할 수 없다고 가정하여 설치 프로그램에서 임의 판별자를 간단히 선택할 수 있습니다.

Pietrzak은 반감 프로토콜을 구현하여 Wesolowski의 RSW 퍼즐을 개선 한 VDF 후보를 제안했습니다. 이는 증명 구성에서 병렬 처리를 허용하고 $\approx\sqrt{t}$ 시간의 감소된 증명 시간을 허용합니다. 그러나 이것은 log(t)의 계수에서 증명 크기와 검증 시간의 대가를 초래합니다.

3. 1-라운드 결제 모델

주문이 만들어진 것과 동일한 라운드($\delta = 0$)에서 주문이 정산되는 모델을 제안합니다. 여기서 결제 논리는 t가 가장 높은 테이크 오더를 결정적으로 선택합니다. 라운드 r에서 r+1까지, 테이커는 라운드가 끝날 때까지 증분 t로 Commit 시퀀스를 계속 제출하고 결제 스마트 계약은 라운드 r+1에서 승리한 테이크 오더를 평가합니다.



위의 다이어그램에서 시작값 $x_{\&}$를 가진 정직한 테이커인 $T_{\&}$와 각각의 시작값 x'를 가진 적대적인 선두 주자 (또는 나중에 동일한 메이크 오더 M을 제출한 충돌하는 정직한 테이커)를 가지고 있습니다. $T_{\&}$는 로컬에서 V (x &)를 계산하고 첫 번째 증명 $\pi_{\&,}$를 제출하고 약간의 지연 t6을 선택합니다. T', $T_{\&}$의 첫 번째 Commit을 관찰한 후 t를 계산하기 위해 경쟁합니다. 그러나 $T_{\&}$의 유리함으로 인해 $T_{\&}$가 최적화되지 않은 하드웨어를 사용하더라도 T'가 $T_{\&}$보다 큰 최종 t를 생성하는 것이 기하 급수적으로 더 어려울 것입니다..

그러나이 모델은 충돌 방지 및 전면 실행 내성이 있습니다. 그렇게하는 것은 경제적으로 비용이 많이 들고 위험할 수 있지만, 공격자는 각 라운드가 시작될 때 오더북 {M6, MF,… M&7}의 모든 주문을 동시에 재계산하고 $T_{\&}$의 첫 번째 Commit(grinding attack)을 관찰한 후 정직한 $T_{\&}$보다 높은 최종 t를 제출할 수 있습니다.

네트워크 노드 또는 채굴자는 $T_{\&}$의 후속 Commit 트랜잭션을 차단한 다음 더 높은 t 값으로 자신의 Commit을 제출하여 $T_{\&}$의 주문을 성공적으로 처리 할 수 있습니다. 그럼에도 불구하고 이 기본 모델은 교환 거래를 결제하고 폐쇄 유동성 re-레이어 뿐만 아니라 이미 전면 실행 저항 조치가 있는 re-레이어에 대한 충돌 저항을 제공하는 데 유용합니다.

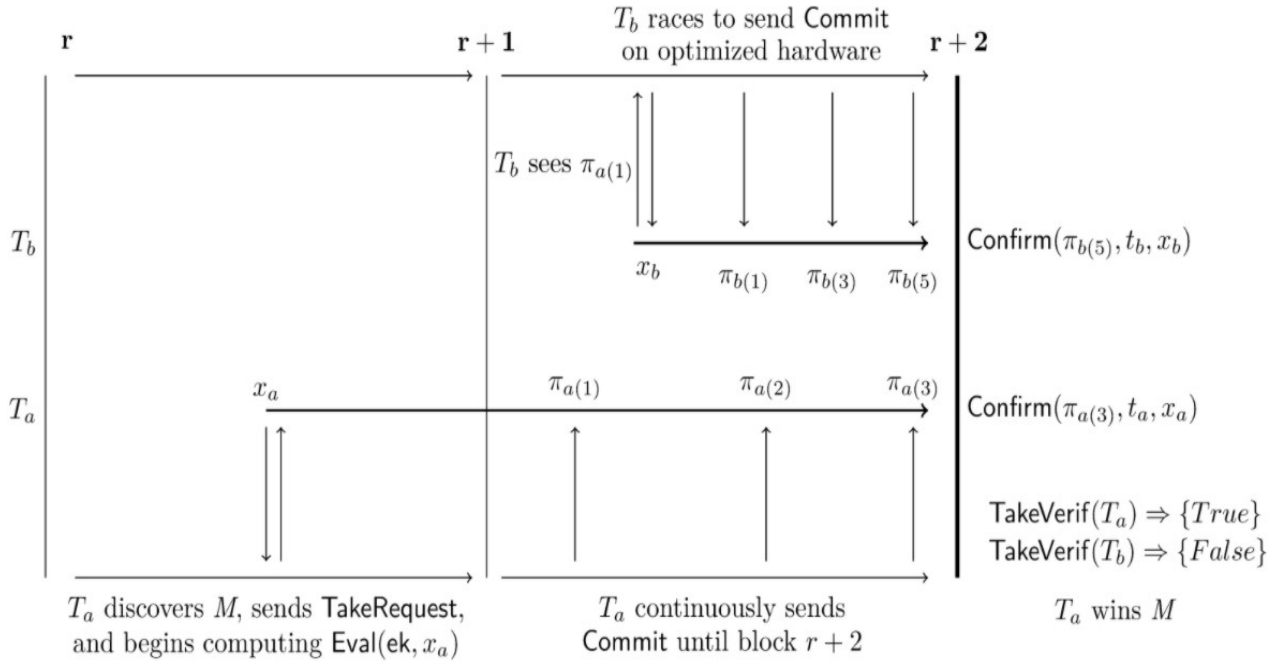
4. 전면 실행 증명 모델

전면 실행에 완전히 대처하기 위해 지연 계수가 $\delta \geq 1$인 Commit 및 공개 방식을 구현하는 모델을 제안합니다.

정의 3.6. 블록 r에서 $T_{\&}$에 의해 결제 스마트 계약에 제출된 테이크 오더 T의 경우, 이전에 정의된대로 xT = H(Sig(sk, m = nonce,-6||T)) = H(S,)입니다. $T_{\&}$는 xT($x_{\&}$로 표시됨)를 제출하여 최소 1 라운드 후 δ 라운드 내에 해당 Commit을 전송하여 T를 확인 (즉, "공개")합니다. 그런 다음 받는 사람은 VDF 구성의 시작 값으로 x&를 사용합니다. 공격자가 모든 조합을 추측하여 해시를 재구성하려고 시도하는 grinding attack을 방지하기 위해 Commit 정보 $x_{\&}$는 메시지 nonce,-6 및 T에 대한 $T_{\&}$의 서명으로 구성됩니다.

정의 3.7. $T_{\&}$는 TakeRequest($x_{\&}$, A) 트랜잭션을 결제 논리에 전송하여 T를 취하겠다는 의도를 제출합니다. δ 블록 시간 프레임 내 이후 블록에서 $T_{\&}$는 TakeVerif(H(S,), $x_{\&}$) → {True, False}로 확인될 Commit을 제출합니다.

아래 다이어그램에는 $\delta = 1$인 모델이 있고 T·는 $T_{\&}$의 순서를 앞 세우려는 적대적인 앞 주자입니다r:



이 시나리오에서 $T_{\&}$는 블록 r에서 TakeRequest를 전송하고 블록 r + 1 동안 Commit을 제출합니다. T'는 $T_{\&}$의 Commit을 보고 T를 발견하고 최적화된 하드웨어를 사용하여 더 높은 경도 매개 변수 t로 Commit을 제출하여 거래를 앞당기려고 합니다. T'가 블록 r에서 TakeRequest를 보내지 않았기 때문에 TakeVerif에 의해 유효하지 않은 것으로 간주되어 t가 더 높았음에도 불구하고 거부되었습니다.

여전히 전면 실행 기회를 갖기 위해 T는 매 라운드마다 오더북의 모든 M에 대해 TakeRequests를 제출할 수 있으며 Commit을 제출하지 않아도 되므로 주문이 공개되지 않고 채워지지 않습니다. 이렇게 하면 상당한 가스 수수료가 발생한다는 사실 외에도, 이 공격은 트레이더가 후속 Commit 없이 대량의 TakeRequest가 제출될 경우 몰수 되는 예금을 하도록 요구함으로써 훨씬 더 많은 비용을 초래할 수 있습니다.

5. 실용적인 고려 사항

두 명의 테이커가 서로 다른 시간에 동일한 TakeRequest를 제출하지만 여전히 동일한 블록 내에 있는 시나리오에서 공정한 시스템은 TakeRequest를 먼저 제출한 테이커가 거래에서 이길 수 있도록 합니다. 비동기화 네트워크에서는 주어진 TakeRequest에 대한 정확한 타임 스탬프를 얻는 것이 불가능하기 때문에 Commit의 t 값을 사용하여 이전 블록에서 누가 TakeRequest를 먼저 제출했는지 판단합니다. 하지만 이 시스템을 통해 최적화된 하드웨어를 사용하는 트레이더는 상용 하드웨어를 사용하는 트레이더가 이전에 TakeRequest를 제출한 경우에도 상용 하드웨어를 사용하는 트레이더를 이길 수 있습니다.

먼저 병렬 프로세서에서 계산할 때 이점을 제공하지 않는 VDF 후보를 선택하여 이 문제를 완화합니다. Sloth++, Pietrzak의 RSW VDF 및 Wesolowski의 RSW VDF는 순차적 특성으로 인해 이 속성을 충족하는 세 가지 후보입니다. 그러나 어떤 VDF 후보가 우리의 설정에 가장 적합한지 아직 명확하지 않습니다.
Wesolowski의 VDF 및 Pietrzak의 RSW에서는 VDF 평가 후에 증명 생성이 발생하는 반면 Sloth ++에서는 증명 생성이 병렬로 계산될 수 있습니다.
또한 증명 생성은 Wesolowski의 RSW에 대해 O(t) 시간이 걸리지만 Pietrzak의 RSW에 대해서는 O(√t) 시간만 걸립니다. 그러나 이러한 속도 향상은 더 긴 검증 시간과 log(t)의 인자에 의해 더 큰 증명 크기를 희생합니다. 사용 사례에 가장 적합한 VDF 후보를 결정하려면 광범위한 테스트가 필요합니다.
경쟁의 장을 맞추기 위해 트레이더는 최적화된 하드웨어에서 실행되는 외부 계산 공급자에게 VDF 계산 및 증명 생성을 위임할 수 있습니다. 그러나 그렇게 하면 중앙 집중화 요소가 도입되고 차선의 사용자 경험을 만들 수 있음을 인지하고 있습니다.

6. 사이드 체인 구현

설정에서 결제 논리는 0x V2 프로토콜을 사용하는 거래 집행 코디네이터(TEC)에 의해 실행됩니다. TEC의 스마트 계약 주소는 모든 주문에 대한 senderAddress 매개 변수가 되며, 이는 TEC가 승인한 주문만 성공적으로 처리하도록 강제합니다. 차례로 TEC 스마트 계약은 이전 섹션에서 설명한 Front-Running Proof Model을 사용하는 사이드 체인을 사용하여 수신 주문의 실제 순서를 설정합니다. 그런 다음 TEC는 0x에서 교환을 수행하여 유효한 것으로 확인된 주문 접수를 허용합니다. 거래 실행을 위해 사이드 체인을 사용함으로써 트레이더는 상당한 가스를 소비하지 않고 사이드 체인에 Commit을 제출할 수 있습니다. 가스 절약 외에 또 다른 이점은 사이드 체인 결제 논리가 프로토콜에 구애 받지 않고 모든 DEX 프로토콜에 대해 공정한 순서를 결정할 수 있다는 것입니다.

또한 사이드 체인의 TEC는 여러 개별 DEXS에서 유동성을 공유할 수 있습니다. 예를 들어, TEC는 IDEX와 같은 다른 분산형 거래소의 주문을 중계하여 공유 유동성을 생성할 수 있습니다. TEC는 유동성 애그리게이터 역할도 할 수있어 트레이더가 여러 거래소에서 여러 오더북에 액세스하고 거래할 수 있습니다..